Anexo 1: Pruebas y construcción del VECM

Toscano Luconi Esquivel

2024-07-04

Introducción al anexo

En este anexo se presenta todo el código de los modelos econométricos estimados, así como las pruebas realizadas.

El flujo de trabajo es el siguiente:

1. Pruebas de Raíz Unitaria.
2. Prueba de Ruido Blanco.
3. Prueba de Cointegración.
4. Prueba de Causalidad de Granger.
5. Estimación del modelo VECM
6. Prueba de Normalidad de los Residuos
7. Prueba de Varianza Constante (Homocedasticidad).
8. Prueba de Autocorrelación.

Se procede a cargar las librerias necesarias para el análisis.

```
# Función para cargar librerias
source("cargar.librerias.r")

# Cargar las librerias necesarias
cargar.librerias("economia")
```

```
## [1] "Cargando librerías de Econometría"
## [1] "Cargadas librerias de Econometría"
```

Se cargan los datos y se presenta un resumen simple de los mismos. Las variables cargadas son:

- TC: Tipo de cambio de dólares americanos por colones costarricenses
- IPC: Índice de Precios al Consumidor
- BM: Base monetaria
- IMAE: Índice Mensual de Actividad Económica
- TPM: Tasa de Política Monetaria
- EXP: Expectativas cambiarias a 12 meses
- RMI: Reservas netas del Banco Central de Costa Rica
- TUR: Cantidad de turistas que ingresaron a Costa Rica

```
# Cargar los datos
datos = read_excel("TipoCambio.xlsx", sheet = 2)

# Revisar los datos
summary(datos[,-c(1,10)])
```

```
##        TC             IPC               BM                IMAE
##  Min.   :526.9   Min.   : 92.34   Min.   : 9620783   Min.   : 91.13
##  1st Qu.:555.2   1st Qu.: 94.17   1st Qu.:12365375   1st Qu.: 98.24
```

```
##  Median :573.5   Median : 98.88   Median :14046972   Median :103.32
##  Mean   :583.5   Mean   : 99.52   Mean   :14445265   Mean   :103.83
##  3rd Qu.:610.7   3rd Qu.:101.72   3rd Qu.:16581014   3rd Qu.:109.26
##  Max.   :692.2   Max.   :113.06   Max.   :19136199   Max.   :121.00
##       TPM             EXP              RMI             TUR
##  Min.   :0.750   Min.   :-0.7278   Min.   : 6192   Min.   :  1022
##  1st Qu.:1.750   1st Qu.: 1.2343   1st Qu.: 7188   1st Qu.:160163
##  Median :3.658   Median : 2.2938   Median : 7792   Median :216661
##  Mean   :3.639   Mean   : 2.2605   Mean   : 7972   Mean   :204094
##  3rd Qu.:5.000   3rd Qu.: 3.1868   3rd Qu.: 8261   3rd Qu.:264824
##  Max.   :9.000   Max.   : 5.5777   Max.   :13219   Max.   :358665
```

Se reduce la variable `datos` para que solo incluya las variables con las que se estima el modelo que busca identificar las variables significativas.

```
dm = datos[,-c(1,10)] # Se eliminan columnas: Fecha y Ventanillas
```

Análisis de Componentes Principales

```
cargar.librerias("analisis")
```

```
  [1] "Cargando librerías de Análisis"
  [1] "Cargadas librerías de Análisis"
```

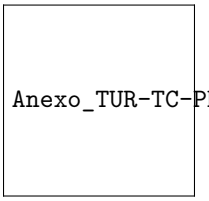
```
# Escalar datos
datos_pca = scale(datos[, -1])

# Estimar PCA
pca_tur = PCA(datos_pca, graph = FALSE, dim(datos_pca)[2])
summary(pca_tur)
```

```
  Call:
  PCA(X = datos_pca, scale.unit = dim(datos_pca)[2], graph = FALSE)


  Eigenvalues
                         Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
  Variance               4.082   2.180   1.195   0.936   0.251   0.209   0.084
  % of var.             45.356  24.226  13.277  10.400   2.794   2.323   0.933
  Cumulative % of var.  45.356  69.582  82.859  93.259  96.052  98.376  99.309
                         Dim.8   Dim.9
  Variance               0.046   0.016
  % of var.              0.512   0.180
  Cumulative % of var.  99.820 100.000

  Individuals (the 10 first)
           Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
  1    |  3.431 | -2.621  1.573  0.584 | -1.913  1.568  0.311 | -0.722  0.408
  2    |  3.215 | -1.968  0.887  0.375 | -2.258  2.186  0.493 | -0.243  0.046
  3    |  3.003 | -1.851  0.784  0.380 | -2.001  1.716  0.444 |  0.288  0.065
  4    |  2.834 | -1.994  0.911  0.495 | -1.481  0.940  0.273 |  0.607  0.288
  5    |  2.707 | -2.072  0.983  0.586 | -1.373  0.809  0.257 |  0.611  0.292
  6    |  2.671 | -1.984  0.901  0.552 | -1.512  0.979  0.320 |  0.654  0.334
  7    |  2.678 | -2.136  1.044  0.636 | -1.100  0.519  0.169 |  0.914  0.654
  8    |  2.788 | -2.284  1.194  0.671 | -0.837  0.301  0.090 |  0.859  0.577
```

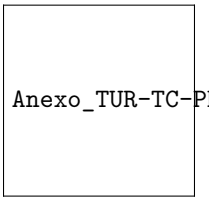
```
  9    |  2.747 | -2.210  1.118  0.647 | -0.677  0.196  0.061 |  1.136  1.009
  10   |  2.690 | -2.217  1.125  0.679 | -0.916  0.360  0.116 |  1.035  0.838
         cos2
  1     0.044 |
  2     0.006 |
  3     0.009 |
  4     0.046 |
  5     0.051 |
  6     0.060 |
  7     0.117 |
  8     0.095 |
  9     0.171 |
  10    0.148 |

  Variables
          Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
  TC    |  0.188  0.868  0.035 |  0.827 31.375  0.684 | -0.487 19.843  0.237 |
  IPC   |  0.950 22.102  0.902 |  0.216  2.135  0.047 | -0.113  1.061  0.013 |
  BM    |  0.867 18.422  0.752 |  0.410  7.702  0.168 |  0.048  0.193  0.002 |
  IMAE  |  0.944 21.845  0.892 |  0.090  0.368  0.008 | -0.046  0.178  0.002 |
  TPM   |  0.627  9.618  0.393 | -0.574 15.127  0.330 | -0.429 15.395  0.184 |
  EXP   | -0.017  0.007  0.000 |  0.522 12.505  0.273 |  0.588 28.889  0.345 |
  RMI   |  0.580  8.230  0.336 | -0.494 11.210  0.244 |  0.560 26.199  0.313 |
  TUR   | -0.025  0.016  0.001 | -0.627 18.033  0.393 | -0.265  5.862  0.070 |
  VEN   |  0.878 18.893  0.771 | -0.184  1.545  0.034 |  0.169  2.381  0.028 |
```

```r
# Valores propios
eig.tmp <- pca_tur$eig                # Seleccionar del modelo
eig.tmp[,2:3]<-eig.tmp[,2:3]/100. # Pasar a porcentaje

# Presentar datos html
DT::datatable(eig.tmp) %>%
  formatRound('eigenvalue',2) %>%
  formatPercentage(c('percentage of variance','cumulative percentage of variance'),2)
```

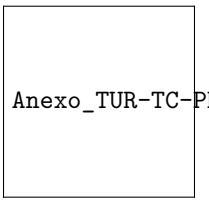
```r
# Tabla de exportación
# stargazer::stargazer(pca_tur$var$cos2)

# Gráfico de varianza explicada adicional
ggplot(data = data.frame(prop_varianza_acum = pca_tur$eig[,3],
                         pc = 1:dim(pca_tur$eig)[1]),
       aes(x = pc, y = prop_varianza_acum, group = 1)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  labs(x = "Componente principal",
       y = "Prop. varianza explicada acumulada")
```



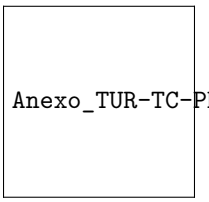
```r
# Presentación de plano principal - individuos
fviz_pca_ind(pca_tur, col.ind="cos2", geom = "point",
             gradient.cols = c("black", "#2E9FDF", "#FC4E07" ),
             title  = "Representación de observaciones en plano principal")
```



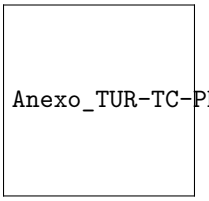

```r
# Presentación de plano principal - vectores
fviz_pca_var(pca_tur, col.var = "cos2",
   gradient.cols = c("black", "blue", "red"))
```




```r
# Presentación de plano principal - combinado
fviz_pca_biplot(pca_tur, col.ind="cos2",col.var = "cos2", geom = "point",
   gradient.cols = c("black", "#2E9FDF", "#FC4E07" ), title  = "Bitplot",repel = TRUE)
```




```r
library(corrplot)
corrplot(pca_tur$var$cos)
```

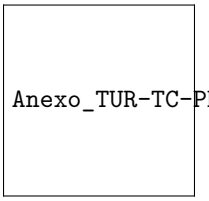


Revisión visual

Partiendo de la elección de las variables para el análisis, se identifica que puede existir un alto grado de colinealidad entre los regresos. Se realiza un inspección visual para detectar cuáles son las variables más correlacionadas a continuación.

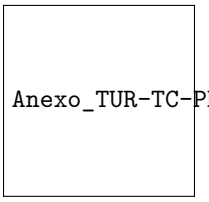
```r
correlacion <- cor(dm)
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlacion, method="color", col=col(200),
         type="lower", order="original",
         addCoef.col = "black",
         tl.col="black", tl.srt=45,
```

```
         diag=FALSE)
```



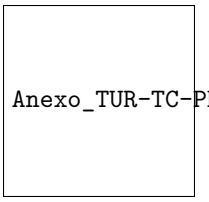
Se deteca con el método anterior que la Base Monetaria tiene una correlación cercana a 0.90 con el IPC (lo que tiene sentido téorico). Asimismo, el IPC y el IMAE tienen una relación del 0.9, que se fundamenta en teoría también; éste último se correlaciona fuertemente con la Base Monetaria adicionalmente. A razón de lo anterior, y fundamentándose en la teoría propuesta por Krugman, se eliminan la Base Monetaria y el IMAE del análisis, dejando entonces solo el nivel de precios - medido por el IPC.

```
dm = dm[,-c(3,4)]
correlacion <- cor(dm)
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlacion, method="color", col=col(200),
         type="lower", order="original",
         addCoef.col = "black",
         tl.col="black", tl.srt=45,
         diag=FALSE)
```



De esta manera no permanecen en los datos relaciones elevadas. Siendo la de las RMI y el TC la más alta (en -0.55) que tiene total congruencia teórica y empírica. Asimismo, se presenta un gráfico donde se muestra la evolución temporal de cada una de las variables elegidas.

```
par(mfrow = c(3,2))
plot(x = datos$Fecha, y = dm$TC, main = "Figura 1. Evolución del tipo de cambio de referencia del BCCR"

plot(x = datos$Fecha, y = dm$TUR, main = "Figura 2. Evolución de llegadas internacionales de turistas a
plot(x = datos$Fecha, y = dm$IPC, main = "Evolución del IPC",
     xlab = "Fecha", ylab = "Nivel (Base = 2020)", type = "l",
     sub = "Elaboración propia con base en datos del BCCR.")
plot(x = datos$Fecha, y = dm$TPM, main = "Evolución de la TPM",
     xlab = "Fecha", ylab = "Tasa", type = "l",
     sub = "Elaboración propia con base en datos del BCCR.")
plot(x = datos$Fecha, y = dm$TC, main = "Evolución del IMAE",
     xlab = "Fecha", ylab = "Nivel (Base = 2017)", type = "l",
     sub = "Elaboración propia con base en datos del BCCR.")
plot(x = datos$Fecha, y = dm$TC, main = "Evolución de las RMI",
     xlab = "Fecha", ylab = "Millones de dólares", type = "l",
     sub = "Elaboración propia con base en datos del BCCR.")
```



Partiendo de la observación anterior se supone la existencia de raíces unitarias en varias de las series.

1: Pruebas de Raíz Unitaria

Las series de tiempo que tienen una tendencia experimentan lo que estadísticamente se conoce como un raíz unitaria. Es decir, cuentan con un valor en su proceso generador de datos que les hace crecer (o decrecer) a través del tiempo. Una serie temporal que no tiene raíz unitaria se puede clasificar como una "caminata aleatoria" y se suele decir que son estacionarias al rededor de un valor (su valor esperado).

La prueba de Dickey-Fuller Aumentada (ADF, por sus siglas en inglés), así como la prueba Phillips-Perron y Kwiatkowski-Phillips-Schmidt-Shin (KPSS), buscan identificar si una serie tiene raíz unitaria o no (es o no es estacionaria). La fomulación específica de cada prueba es difente, aunque se busca probar lo mismo. Por eso se presenta la estructura de la ADF como referencia.

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \cdots + \delta_p \Delta y_{t-p} + \varepsilon_t$$

donde:

- Δy_t es la primera diferencia de la serie temporal
- α es una constante (intercepto).
- βt es el término de tendencia temporal (si se incluye).
- $\$\gamma\$eselcoeficientedelvalordelaserieenelperiodopasado.\delta_i$ son los coeficientes de las diferencias rezagas de la serie.
- $varepsilon_t$ es el término de error estocástico.

La hipótesis de la prueba es la siguiente:

$$\begin{aligned} H_0 &: \gamma = 0 \quad \text{(La serie tiene una raíz unitaria, indicando no estacionariedad)} \\ H_1 &: \gamma < 0 \quad \text{(La serie no tiene una raíz unitaria, indicando estacionariedad)} \end{aligned}$$

Con base en la teoría expuesta, se estiman las pruebas para cada serie y se reporta a continuación.

```
# Crear un data frame vacío con el número correcto de filas y sin nombres de fila inicialmente
p_values = data.frame(ADF = numeric(length(names(dm))), PP = numeric(length(names(dm))), KPSS = numeric

# Iterar sobre cada columna en el data frame 'dm'
for (variable in names(dm)) {
  # Prueba ADF
  adf_test = adf.test(dm[[variable]], alternative = "stationary", k = trunc((length(dm[[variable]]) - 1

  # Prueba PP
  pp_test = pp.test(dm[[variable]], alternative = "stationary")

  # Prueba KPSS (asumiendo que se prueba la estacionariedad en torno a una tendencia)
  kpss_test = tryCatch(kpss.test(dm[[variable]], null = "Trend"),
                       warning = function(w) return(list(statistic = NA, p.value = NA)),
                         error = function(e) return(list(statistic = NA, p.value = NA)))

  # Almacenar p-values en el data frame de resultados
  p_values[variable, "ADF"] = adf_test$p.value
```

```
  p_values[variable, "PP"] = pp_test$p.value
  p_values[variable, "KPSS"] = kpss_test$p.value
}

# Asignar nombres de fila después de crear el data frame
rownames(p_values) <- names(dm)

# Presentar el data frame de p-values
print(round(p_values,2))
```

```
      ADF   PP KPSS
  TC  0.77 0.94 0.02
  IPC 0.33 0.72   NA
  TPM 0.33 0.79 0.01
  EXP 0.11 0.41 0.04
  RMI 0.99 0.99 0.01
  TUR 0.34 0.08 0.04
```

Según lo que se observa en la tabla anterior, todas las series presentan raíz unitaria. La más cercana a serlo es EXP. Nota: la prueba KPSS es inversa a la de la ADF y PP en cuanto a la hipótesis. Para corregir la raíz unitaria lo que se requiere es diferenciar la seria hasta hacerla estacionaria; en este caso se supone que con un orden $I(1)$ se puede encontrar que todas las variables son estacionarias.

```
# Crear un nuevo data frame vacío con la misma estructura que 'dm'
dm_dif = data.frame(matrix(ncol = ncol(dm), nrow = nrow(dm) - 1))

# Asignar los nombres de las columnas de 'dm' a 'dm_dif'
names(dm_dif) = names(dm)

# Iterar sobre cada columna en 'dm' para calcular la primera diferencia
for (variable in names(dm)) {
  dm_dif[[variable]] = diff(dm[[variable]])
}

# Asignar nombres de fila
rownames(dm_dif) = tail(rownames(dm), -1)
```

Se realiza entonces de nuevo la prueba para verificar la estacionariedad.

```
# Crear un data frame vacío con el número correcto de filas y sin nombres de fila inicialmente
p_values_dif = data.frame(ADF = numeric(length(names(dm_dif))), PP = numeric(length(names(dm_dif))), KPS

# Iterar sobre cada columna en el data frame 'dm'
for (variable in names(dm_dif)) {
  # Prueba ADF
  adf_test <- adf.test(dm_dif[[variable]], alternative = "stationary", k = trunc((length(dm_dif[[variabl

  # Prueba PP
  pp_test = pp.test(dm_dif[[variable]], alternative = "stationary")

  # Prueba KPSS (asumiendo que se prueba la estacionariedad en torno a una tendencia)
  kpss_test = tryCatch(kpss.test(dm_diff[[variable]], null = "Trend"),
                       warning = function(w) return(list(statistic = NA, p.value = NA)),
                       error = function(e) return(list(statistic = NA, p.value = NA)))
```

```
  # Almacenar p-values en el data frame de resultados
  p_values_dif[variable, "ADF"] = adf_test$p.value
  p_values_dif[variable, "PP"] = pp_test$p.value
  p_values_dif[variable, "KPSS"] = kpss_test$p.value
}

# Asignar nombres de fila después de crear el data frame
rownames(p_values_dif) <- names(dm_dif)

# Presentar el data frame de p-values
print(round(p_values_dif,2))
```

```
      ADF   PP KPSS
  TC  0.03 0.01   NA
  IPC 0.20 0.01   NA
  TPM 0.18 0.01   NA
  EXP 0.02 0.01   NA
  RMI 0.01 0.01   NA
  TUR 0.01 0.01   NA
```

Se obtiene que ahora las series `TC`. `EXP`, `RMI` y `TUR` son estacionarias, mientras que `IPC` y `TPM` necesitarían una segunda diferencia para serlo. Por motivo de interpretabilidad se toma la decisión de no diferenciarlas y estimar dos modelos: uno con las variables, asumiendo el riesgo de efectos sesgados, y otro sin las variables.

2: Pruebas de Ruido Blanco

Habiendo transformado las series en procesos estacionarios, es necesario realizar la prueba de Ruido Blanco. La prueba de Ljung-Box es una forma de probar la hipótesis nula de que los datos en una serie temporal son independientes, es decir, no autocorrelacionados hasta un número de lags k. La hipótesis nula y alternativa se pueden expresar como:

H_0 : Los datos no muestran autocorrelación hasta el lag k.

H_1 : Los datos muestran autocorrelación en al menos uno de los lags hasta k.

El estadístico de prueba Q se calcula de la siguiente manera:

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k} \rightarrow \chi_h^2$$

donde:

- n es el tamaño de la muestra,
- $\hat{\rho}_k$ es la autocorrelación estimada en el lag k,
- h es el número total de lags a considerar.

```
white_noice = data.frame(WN = numeric(length(names(dm_dif))),
                         row.names = names(dm_dif))

# Iterar sobre cada columna en el data frame 'dm'
for (variable in names(dm_dif)) {
  # Prueba de Ruido Blanco
  wn_test = Box.test(dm_dif[[variable]], lag = 12,
                     type = c("Ljung-Box"), fitdf = 0)

  # Almacenar p-values en el data frame de resultados
```

```
  white_noice[variable, "WN"] = wn_test$p.value
}

# Asignar nombres de fila después de crear el data frame
rownames(white_noice) = names(dm_dif)

# Presentar el data frame de p-values
print(round(white_noice,2))
```

```
       WN
  TC  0.01
  IPC 0.00
  TPM 0.00
  EXP 0.15
  RMI 0.00
  TUR 0.00
```

Según lo que se observa en la tabla anterior, solo la serie `EXP` parece ser ruido blanco. Esto es interesante dado que la serie es una observación de la expectativa de variación cambiaria promedio realizada mediate una encuesta a diversos economistas en Costa Rica.

3: Pruebas de Cointegración

Sabiendo que las series no son ruido blanco, se procede a identificar si existe o no cointegración entre ellas. Como ya se identificó que las pruebas tienen raiz unitaria, se sabe que existe un vector de cointegración que las haría estacionarias.

```
dm_dif = dm_dif[,-c(2,3)] # Se eliminan las variables IPC y TPM
rezagos = VARselect(dm_dif, lag.max = 12) # Identifación de rezagos
rezagos$selection # Selección de rezagos
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      2      1      4
```

```
vec_coint = ca.jo(dm_dif[,-2], type="trace", ecdet="none", K=2, spec="longrun") # Se deja fuera EXP.
summary(vec_coint) # Hay al menos dos relaciones de cointegración
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.4400971 0.3772073 0.1506018
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  16.98  6.50  8.18 11.65
## r <= 1 |  66.22 15.66 17.95 23.52
## r = 0  | 126.54 28.71 31.52 37.22
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
```

```
##
##                TC.12       RMI.12        TUR.12
## TC.12   1.0000000000 1.0000000000  1.000000e+00
## RMI.12  0.0581374837 0.0417396628 -3.610526e-03
## TUR.12 -0.0008511946 0.0001385891 -1.246968e-05
##
## Weights W:
## (This is the loading matrix)
##
##               TC.12        RMI.12      TUR.12
## TC.d    0.001250966   -0.05823913  -0.4950952
## RMI.d  -2.903338835  -18.01308372   4.8343828
## TUR.d 897.890991852 -931.73177995 248.9449227
```

4: Pruebas de Causalidad de Granger

Siguiendo con el flujo de trabajo, se genera a través del código una iteración para determinar la causalidad de Granger entre las variables. En esta construcción se hace la revisión bi-direccional, entonces cada variable i es contrastada con la variable j y vice versa.

```
granger_tests = function(data, var1, var2, max_lag = 2) {
  test1 = grangertest(data[[var2]] ~ data[[var1]], order = max_lag, data = data)
  test2 = grangertest(data[[var1]] ~ data[[var2]], order = max_lag, data = data)

  list(
    test = paste(var1, "Granger-causes", var2),
    test2 = paste(var2, "Granger-causes", var1)
  )
}
```

```
# Crear Data Frames para guardar las relaciones
variable_names = names(dm_dif)
granger_results = list()

# Iterar sobre todas las combinaciones posibles de variables
for (i in 1:(length(variable_names) - 1)) {
  for (j in (i + 1):length(variable_names)) {
    var1 = variable_names[i]
    var2 = variable_names[j]

    # Ejecutar la prueba de Granger para ambos pares y direcciones
    results = granger_tests(dm_dif, var1, var2, max_lag = 2)
    granger_results[[paste(var1, "and", var2)]] = results
  }
}

# Para imprimir y revisar los resultados
print(granger_results)
```

```
  $`TC and EXP`
  $`TC and EXP`$test
  [1] "TC Granger-causes EXP"

  $`TC and EXP`$test2
  [1] "EXP Granger-causes TC"
```

```
$`TC and RMI`
$`TC and RMI`$test
[1] "TC Granger-causes RMI"

$`TC and RMI`$test2
[1] "RMI Granger-causes TC"


$`TC and TUR`
$`TC and TUR`$test
[1] "TC Granger-causes TUR"

$`TC and TUR`$test2
[1] "TUR Granger-causes TC"


$`EXP and RMI`
$`EXP and RMI`$test
[1] "EXP Granger-causes RMI"

$`EXP and RMI`$test2
[1] "RMI Granger-causes EXP"


$`EXP and TUR`
$`EXP and TUR`$test
[1] "EXP Granger-causes TUR"

$`EXP and TUR`$test2
[1] "TUR Granger-causes EXP"


$`RMI and TUR`
$`RMI and TUR`$test
[1] "RMI Granger-causes TUR"

$`RMI and TUR`$test2
[1] "TUR Granger-causes RMI"
```

5: Estimacion del VECM

Estimación directa

```r
# Construcción del modelo
vecm = ca.jo(dm_dif[,-2], type="eigen", ecdet="none", K=2, spec="longrun")
summary(vecm)
```

```
######################
# Johansen-Procedure #
######################
```

```
Test type: maximal eigenvalue statistic (lambda max) , with linear trend

Eigenvalues (lambda):
[1] 0.4400971 0.3772073 0.1506018

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 | 16.98  6.50  8.18 11.65
r <= 1 | 49.25 12.91 14.90 19.19
r = 0  | 60.32 18.90 21.07 25.75

Eigenvectors, normalised to first column:
(These are the cointegration relations)

               TC.12       RMI.12        TUR.12
TC.12   1.0000000000 1.0000000000  1.000000e+00
RMI.12  0.0581374837 0.0417396628 -3.610526e-03
TUR.12 -0.0008511946 0.0001385891 -1.246968e-05

Weights W:
(This is the loading matrix)

              TC.12        RMI.12      TUR.12
TC.d    0.001250966   -0.05823913  -0.4950952
RMI.d  -2.903338835  -18.01308372   4.8343828
TUR.d 897.890991852 -931.73177995 248.9449227
```

```
# Ajuste del modelo VECM con los rangos de cointegración apropiados
vecm_fit = cajorls(vecm, r = 2)  # r es el número de relaciones de cointegración
vecm_fit
```

```
$rlm

Call:
lm(formula = substitute(form1), data = data.mat)

Coefficients:
          TC.d        RMI.d       TUR.d
ect1      -5.699e-02  -2.092e+01  -3.384e+01
ect2      -2.358e-03  -9.207e-01   1.331e+01
constant   1.515e-02   3.979e+01   1.144e+02
TC.dl1    -7.160e-01  -1.957e+01  -8.333e+01
RMI.dl1    1.394e-03  -7.320e-01  -1.273e+01
TUR.dl1   -1.154e-05   9.221e-04  -4.628e-01


$beta
              ect1        ect2
TC.12  1.000000000  0.00000000
RMI.12 0.000000000  1.00000000
TUR.12 0.002658024 -0.06036069
```

Se utiliza la siguiente libreria para obtener los resultados en un formato más presentable.

```r
library(tsDyn)
```

```
##
## Attaching package: 'tsDyn'
```

```
## The following object is masked from 'package:fabletools':
##
##     MAPE
```

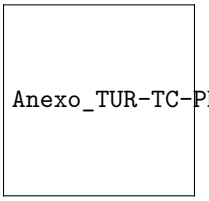
```r
vecm_nuevo = VECM(dm_dif[,-2], lag = 1, r = 2, estim = "ML")
summary(vecm_nuevo)
```

```
## ##############
## ###Model VECM
## ##############
## Full sample size: 106    End sample size: 104
## Number of variables: 3   Number of estimated slope parameters 18
## AIC 3884.816     BIC 3937.704    SSR 164909449452
## Cointegrating vector (estimated by ML):
##    TC RMI          TUR
## r1  1   0  0.002658024
## r2  0   1 -0.060360689
##
##
##              ECT1                    ECT2
## Equation TC  -0.0570(0.0823)         -0.0024(0.0036)
## Equation RMI -20.9164(2.8035)***     -0.9207(0.1219)***
## Equation TUR -33.8408(365.2221)      13.3110(15.8797)
##              Intercept                TC -1
## Equation TC  0.0151(0.9208)           -0.6590(0.0910)***
## Equation RMI 39.7854(31.3520)         1.3440(3.0998)
## Equation TUR 114.4452(4084.2741)      -49.4866(403.8164)
##              RMI -1                TUR -1
## Equation TC  0.0038(0.0028)        -2.4e-06(2e-05)
## Equation RMI 0.1886(0.0938)*       0.0009(0.0007)
## Equation TUR -26.0418(12.2136)*    0.4306(0.0909)***
```

```r
# stargazer::stargazer(vecm_nuevo$coefficients) Output para LaTeX
```

Tranformación a VAR para pruebas

```r
var_final = vec2var(vecm, r = 2)

impulso = irf(var_final, impulse = "TUR", response = "TC", n.ahead = 12, ortho = TRUE, runs = 1000)

plot(impulso, xlab = "Periodos", ylab = "Movimiento de TC", main = "Figura 4. Respuesta del TC ante impu
```



Anexo_TUR-TC-PDFGEN_files/figure-latex/Transformacion-1.pdf

6: Prueba de Normalidad

Con base en los resultados obtenidos en la estimación del VECM, se realiza la prueba de normalidad de los residuos. Los estadísticos de las distintas pruebas presentan evidencia en contra de la hipótesis de normalidad de los residuos.

```
# Normalidad
norm_results = normality.test(var_final, multivariate.only = TRUE)
norm_results
```

```
$JB

    JB-Test (multivariate)

data:  Residuals of VAR object var_final
Chi-squared = 354.74, df = 6, p-value < 2.2e-16


$Skewness

    Skewness only (multivariate)

data:  Residuals of VAR object var_final
Chi-squared = 94.199, df = 3, p-value < 2.2e-16


$Kurtosis

    Kurtosis only (multivariate)

data:  Residuals of VAR object var_final
Chi-squared = 260.54, df = 3, p-value < 2.2e-16
```

7: Prueba de Homocedasticidad

En cuanto a la heterocedasticidad de los residuos, un p-value menor a 0.05 en la prueba ARCH indica que existe evidencia suficiente para rechazar la hipótesis nula de que los residuos presentan una distribución normal, es decir, son homocedásticos.

```
# Heterocedasticidad
arch_results = arch.test(var_final, lags.multi = 12, multivariate.only = TRUE)
arch_results
```

```
    ARCH (multivariate)

data:  Residuals of VAR object var_final
Chi-squared = 485.18, df = 432, p-value = 0.03901
```

8: Prueba de Autocorrelación

Finalmente, la prueba de autocorrelación de los residuos evidencia que si existe correlación entre ellos.

```
# Autocorrelación Seriel
serial_results = serial.test(var_final, lags.pt = 2, type = "PT.asymptotic")
```

```
serial_results
```

```
    Portmanteau Test (asymptotic)

  data:  Residuals of VAR object var_final
  Chi-squared = 9.2648, df = 3, p-value = 0.02597
```

Simulación de significancia del estimador

En este apartado se genera una simulación con base en los estadísticos obtenidos en la estimación del VECM; especialmente se toma el coeficiente y el error estándar de la variable de interés. Lo anterior se realiza con la intención de probar si el valor obtenido del coeficiente puede llegar a ser significativo para el análisis.

El resultado final arroja que lo anterior no se cumple a nivel estadístico. El coeficiente estimado y su respectivo intervalo son congruentes aun al desarrollar miles de simulaciones con el modelo de Monte Carlo.

```
# Datos dados
coeficiente <- -0.0000024
error_estandar <- 0.00002

# Calcular el valor t
valor_t <- coeficiente / error_estandar

# Calcular el valor p usando la distribución t de Student
grados_libertad <- 106 - 15
valor_p <- 2 * pt(-abs(valor_t), df = grados_libertad)

# Imprimir los resultados
print(paste("Valor t: ", valor_t))
```

```
  [1] "Valor t:  -0.12"
```

```
print(paste("Valor p: ", valor_p))
```

```
  [1] "Valor p:  0.904747687899542"
```

```
# Definir los valores del coeficiente y el error estándar
coeficiente <- -0.0000024
error_estandar <- 0.00002

# Número de simulaciones
n_simulaciones <- 1000

# Almacenar los valores t
valores_t <- numeric(n_simulaciones)

# Realizar las simulaciones
set.seed(123)  # Para reproducibilidad
for (i in 1:n_simulaciones) {
    # Simular el coeficiente como una muestra normal con media = coeficiente y sd = error estándar
    coef_simulado <- rnorm(1, mean = coeficiente, sd = error_estandar)

    # Calcular el valor t
    t_simulado <- coef_simulado / error_estandar
    valores_t[i] <- t_simulado
```

```r
}

# Calcular el número de veces que el valor t es estadísticamente significativo
# Usamos un nivel de significancia del 5% para dos colas
t_critico <- qt(0.975, df = 999)  # Grados de libertad aproximados como n_simulaciones - 1
significativos <- sum(abs(valores_t) > t_critico)

# Calcular el porcentaje de resultados significativos
porcentaje_significativos <- significativos / n_simulaciones * 100

# Mostrar resultados
cat("Número de veces que el valor t es significativo:", significativos, "\n")
```

```
  Número de veces que el valor t es significativo: 57
```

```r
cat("Porcentaje de veces que el valor t es significativo:", porcentaje_significativos, "%\n")
```

```
  Porcentaje de veces que el valor t es significativo: 5.7 %
```